ISSN 1561-5359. Штучний інтелект, 2016, № 2

УДК 519.6

А.М. Терещенко, В.К. Задірака
Інститут кібернетики ім. В.М. Глушкова НАН України,
проспект Академіка Глушкова, 40, Київ, 03178

ШВИДКЕ ОБЧИСЛЕННЯ ЦИКЛІЧНОЇ ЗГОРТКИ БАГАТОРОЗРЯДНИХ ЧИСЕЛ НА ОСНОВІ ШПФ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕНЬ

A.M. Tereshchenko, V.K.Zadiraka
VM Glushkov Institute of Cybernetics of NAS of Ukraine
40 Glushkov ave., Kyiv, Ukraine, 03187

FAST PROCEEDING OF CYCLIC CONVOLUTION OF MULTIDIGIT VALUES BASED ON FFT IN PARALLEL COMPUTATIONAL MODEL

Аналізується складність за кількістю однослівних операцій при реалізації операції ЦЗ (циклічної згортки) у паралельній моделі обчислень. Розглянуто методи обчислення ЦЗ, коли кожна точка згортки є багаторозрядним числом. Запропоновано швидкий метод обчислення ЦЗ такого виду на основі ШПФ невеликої довжини.
Ключові слова: багаторозрядне множення, паралельна модель обчислень, циклічна згортка, ШПФ.

The complexity of number of single precision operations is analyzed in multidigit convolution computation in parallel computational model. Calculation methods of cyclic convolution elements are considered when every element is high precision value. The effective method based on FFT of small length of calculation of cyclic convolution is proposed.
Key words: multidigit multiplication, parallel computational model, cyclic convolution, FFT.

Вступ

У багатьох задач цифрової обробки (ЦО) необхідно обчислювати циклічну згортку (ЦЗ). Деякі задачі ЦО можна звести до задачі обчислення ЦЗ. Оптимізації обчислення ЦЗ приділено багато уваги [1, 2]. При обчисленні згорток великої довжини (більше 1024 точок) з використанням звичайної арифметики з плаваючою комою одинарної (або подвійної) точності накопичується похибка заокруглення. Якщо похибка заокруглення є великою, то вона впливає на результат обчислення так, що результат обчислення втрачає достовірність [3]. Для уникнення похибки заокруглення, необхідно оперувати більшою кількістю розрядів у кожній операції. Використання операцій з більшою розрядністю значно уповільнює виконання багаторозрядної операції.

Сучасні процесори дозволяють виконувати операції над числами, довжина яких не перевищує 64 біти (подвійна точність) на апаратному рівні. Роботи [4, 5] присвячені оптимізації обчислення ЦЗ, але з точністю операцій процесора. Для отримання більшої точності, необхідно використовувати спеціальну арифметику, реалізовану на програмному рівні.

Окрім ЦО, згортка використовується для реалізації операцій багаторозрядної арифметики [3]. При реалізації криптографічних операцій найбільше навантаження лягає на операцію багаторозрядного множення. У роботах [4, 5] показано метод обчислення операції множення на основі ЦЗ. Найбільш швидкий метод реалізації багаторозрядної операції множення базується на ШПФ (швидкому перетворенні Фур‘є).

Метод множення «у стовпчик» є ефективним при множенні багаторозрядних чисел з кількістю слів $M < 256$ (де кожне слово має довжину у 32 біти). Метод ШПФ є ефективним, починаючи з $M \geq 256$. Операція множення двох чисел довжиною $M = 256$ слів може бути реалізована на основі ШПФ довжиною $2M = 512$ (метод Кулі-Туки).

При цьому треба враховувати, що при обчисленні ОШПФ (оберненого швидкого перетворення Фур‘є) необхідно виконувати ділення на $2M = 512$. При діленні на 512 відбувається зсув на $\log_2 512 = 9$ бітів. В операціях одинарної точності (32 біта) мантиса займає 23 біти. Тобто, після ділення на 512 (або зсуву вправо на 9 бітів) залишається $14 = 23 - 9$ значущих бітів, що говорить про те, що на початку обчислень враховується тільки 7 значущих бітів, бо при кожному множенні довжина результату подвоюється, а кожне подальше додавання може збільшувати результат на один біт. Так, наприклад, при використанні 8 значущих бітів на початку обчислень після першого множення результат буде займати 16 значущих бітів. Якщо обчислення відбувається, наприклад, для обчислення ЦЗ довжиною 256 (8 бітів) точок, то після 8-ої ітерації ШПФ кількість заповнених значущих бітів буде більшою, ніж 24.

При використанні операцій подвійної точності (64 біти) довжина мантиси складає 51 біт. Тобто, на початку обчислення ШПФ кількість значущих бітів не повинна перевищувати $21 = (51 - 9)/2$ при множенні 256-слівного числа на основі ШПФ довжиною 512. Зрозуміло, що при множенні чисел довжиною більшою, ніж 256 слів, кількість значущих бітів повинна бути меншою, ніж 21, для уникнення переповнення.

Виходячи з попередніх міркувань, множення на основі ШПФ є найшвидшим методом, але існує апаратне обмеження довжини множників у $256 \cdot 21 = 5376$ бітів, які можна перемножити на основі ШПФ, використовуючи операції подвійної точності. При реалізації множення чисел більших, ніж 5376 бітів, на основі ШПФ, мантиса повинна мати більше ніж 51 біт для уникнення переповнення. Виникає потреба використання арифметики з точністю, більшою ніж подвійна, яку необхідно реалізовувати програмно. На практиці це виглядає так, що багаторозрядне число ділиться на M секцій, де кожна секція займає більше ніж 128 бітів. Довжина ШПФ, на основі якої виконується обчислення, дорівнює числу секцій N. Можна використовувати різні методи обчислення для кожного «рівня». Один метод використовувати для реалізації багаторозрядної операції множення для M (множник з M секцій), а інший метод – для оперування розрядами для N (кожна секція складається з N слів, кожне з яких має 32 біти). Метод на основі ШПФ буде найефективнішим при довжинах, більших ніж 256 секцій (для рівня M) та 256 слів (для рівня N). Для довжин, менших ніж 256, для рівнів M та N метод на основі ШПФ програє методу множення «у стовпчик».

Загальноприйнято, що ШПФ треба використовувати, коли довжина ШПФ перевищує 256 точок. У даній роботі показано, що використання ШПФ довжиною 16 є ефективним при обчисленні ЦЗ великих чисел. Запропонований метод дає меншу складність за кількістю однослівних операцій у порівнянні з методом «у стовпчик».

Одним з методів прискорення обчислення багаторозрядних операцій є їх реалізація у паралельній моделі обчислення, окрім використання швидких методів, які є ефективними у своїх діапазонах довжин багаторозрядних чисел. Може здаватися, що при задіянні N паралельних процесорів, можна прискорити час виконання у N разів. Але, на практиці, це зовсім не так. По-перше, не всі методи можна розпаралелити. Так, деякі методи мають пов'язані кроки обчислення, коли наступний крок обчислення оснований на результаті попереднього і не може бути виконаний незалежно від інших кроків. По-друге, при більшій кількості задіяних процесорів необхідно враховувати більше обмежень. Так, наприклад, розмір кеш-пам'яті є постійним і збільшення кількості процесорів означає, що кожному процесору буде виділено менше кеш-пам‘яті. При розбитті процесорів у групи, необхідно виконувати синхронізацію між групами, що є дуже витратною операцією. При ще більшому збільшенні кількості

процесорів необхідно враховувати, що кожен паралельний процесор потребує енергію.

Далі у роботі багаторозрядними числами будемо називати числа, які займають більше ніж два слова у пам'яті комп'ютера. Великі цілі числа є багаторозрядними числами. Далі у роботі використовується термін багаторозрядні числа без розділення на великі цілі числа та числа з плаваючою комою з великою мантисою. Отримані у роботі результати можуть бути використані як для оптимізації обчислень з великими цілими числами, так і для реалізації операцій для отримання більшої кількості розрядів після коми.

Мета дослідження

Метою даної роботі є знаходження швидкого алгоритму обчислення циклічної згортки багаторозрядних чисел у паралельній моделі обчислення. Швидкий алгоритм має бути ефективним, використовувати максимально кеш-пам'ять та векторні операції і задіяти невелику кількість паралельних процесорів для зменшення обсягу використання електроенергії при реалізації у паралельній моделі обчислень. Для цього проведено аналіз складності реалізації операції циклічної згортки багаторозрядних чисел на основі двох методів реалізації багаторозрядної операції множення: множення «у стовпчик» та на основі ШПФ. Досліджено область диференційованої поведінки алгоритмів на основі двох методів. Показано, що запропонований метод на основі ШПФ невеликої довжини є ефективним для умов, коли багаторозрядне число займає не менше восьми слів.

Постановка задачі

Нехай багаторозрядні числа X_i, Y_i, $i=\overline{0,M-1}$ складаються з M секцій. Кожна i секція є N-слівним числом, у свою чергу, де кожне число має довжину у ω-біт. Необхідно побудувати швидкий алгоритм обчислення циклічної згортки довжиною M секцій, де кожна секція є N-слівним числом, у паралельній моделі обчислень:

$$R_j \leftarrow \sum_{i=0}^{M-1}(X_i \cdot Y_{\langle i+j\rangle_M}),\ j=\overline{0,M-1}, \tag{1}$$

де R_j – $2N$-слівне число.

Наприклад, обчислення ЦЗ $R_4 = X_4 \otimes Y_4$ довжиною $M=4$ розрядів можна представити у наступному вигляді:

X_0	Y_0	Y_1	Y_2	Y_3
X_1	Y_1	Y_2	Y_3	Y_0
X_2	Y_2	Y_3	Y_0	Y_1
X_3	Y_3	Y_0	Y_1	Y_2
	R_0	R_1	R_2	R_3

Рис. 1. Обчислення ЦЗ довжиною у 4 точки

Вираз (1) може бути обчислений, використовуючи метод «у стовпчик» при реалізації багаторозрядної операції множення. Розглянемо складність такої реалізації у паралельній моделі обчислень.

еалізація на основі множення «у стовпчик»

Знайдемо спочатку складність алгоритму множення двох N-слівних чисел $X \cdot Y$ стандартним методом «у стовпчик». Для спрощення у наступному алгоритмі (і далі по тексту) вважаємо, що знак переносу у $2N+1$ слово не виникає. Множення «у стовпчик» двох N-слівних чисел $X \cdot Y$ можна представити у вигляді наступної

формули:

$$R=\sum_{k=0}^{2N-1}(r_k\cdot 2^{\omega k}),\ r_k=\begin{Bmatrix}\sum\limits_{t=0}^{k}(x_k\cdot y_{k-t}) & k<N\\ \sum\limits_{t=k-N+1}^{N-1}(x_t\cdot y_{k-t}) & k\geq N\end{Bmatrix},$$

де R - $2N$-слівне число.

Використаємо наступні позначення: $H(S)=s_1, L(S)=s_0$ – знаходження старшої та молодшої частин двослівного числа $S=s_1\cdot 2^{\omega}+s_0$.

Алгоритм 1. Множення N-слівних чисел $X\cdot Y$ стандартним методом «у стовпчик» при задіянні N-паралельних процесорів без врахування знаків переносу.

На вході: $X=\sum\limits_{k=0}^{N-1}(x_k\cdot 2^{\omega k})$, $Y=\sum\limits_{k=0}^{N-1}(y_k\cdot 2^{\omega k})$, p – індекс процесора.

На виході: $R=\sum\limits_{k=0}^{2N-1}(r_k\cdot 2^{\omega k})$ – результат множення $X\cdot Y$.

Крок 1. $r_{2p}\leftarrow r_{2p+1}\leftarrow 0$. // Ініціалізація

Крок 2. Для $k=0$ по $N-1$

Крок 3. $S\leftarrow S+x_p\cdot y_k$ // Множення

Крок 4. $r_{p+k}\leftarrow r_{p+k}+L(S)$, $r_{p+k+1}\leftarrow r_{p+k+1}+H(S)$. // Додавання

Крок 5. Кінець циклу по k.

Примітка. Алгоритм 1 не є оптимальним за кількістю операцію запису у пам'ять. Головною метою Алгоритму 1 є представлення загальної кількості операцій, необхідних для реалізації багаторозрядної операції.

Лема 1. В Алгоритмі 1 при множенні двох N-слівних чисел стандартним методом «у стовпчик» кожен з N паралельних процесорів виконає N однослівних операцій множення та $2N$ однослівних операцій додавання.

Доведення. На кроках 2-5 необхідно виконати N ітерацій циклу. З врахуванням того, що результатом множення двох однослівних чисел є двослівне число, необхідно виконати у два рази більшу кількість однослівних операцій додавання. Лема доведена.

Далі розглянемо алгоритм, який обчислює циклічну згортку (1), при задіянні MN паралельних процесорів. Алгоритм знаходить суми добутків N-слівних чисел $X_i\cdot Y_{\langle i+j\rangle_M}$, $i=\overline{0,M-1}$, $j=\overline{0,M-1}$, на основі стандартного методу множення «у стовпчик».

Алгоритм 2. Знаходження суми добутків N-слівних чисел на основі методу множення «у стовпчик» при задіянні MN паралельних процесорів без врахування знаків переносу.

На вході: X_i, Y_i, $i=\overline{0,M-1}$;

$j=\overline{0,M-1}$ – індекси ряду та стовпчика групи процесорів, яку можна інтерпретувати у вигляді матриці.

На виході: $R_j=\sum\limits_{i=0}^{M-1}(X_i\cdot Y_{\langle i+j\rangle_M})$, $j=\overline{0,M-1}$.

Крок 1. $R_j \leftarrow 0$. // Ініціалізація

Крок 2. Для $i = 0$ по $M-1$

Крок 3. $R_j \leftarrow R_j + X_i \cdot Y_{\langle i+j \rangle_M}$. // N -слівне множення

Крок 4. Кінець циклу по i. // на основі Алгоритму 1

Лема 2. В Алгоритмі 2 при знаходженні сум множень N -слівних чисел на основі методу множення «у стовпчик» кожен з MN паралельних процесорів виконає MN однослівних операцій множення та $2MN$ однослівних операцій додавання.

Доведення. Згідно з Лемою 1 при множення двох N-слівних чисел необхідно виконати N однослівних операцій множень та $2N$ однослівних операцій додавання. Таких ітерацій необхідно виконати M . Лема доведена.

Реалізація на основі ШПФ

У даній роботі пропонується обчислювати ЦЗ (1) великих чисел, використовуючи метод множення на основі ШПФ, у паралельній моделі обчислень.

Розглянемо алгоритм множення N -слівних чисел $X \cdot Y$ на основі ШПФ довжини $2N$. Обчислення на основі ШПФ можна представити наступним чином:

$$R = X \cdot Y = \frac{1}{2N} \cdot W^*_{2N,2N} \cdot ((W_{2N,2N} \cdot (X, Z)) \cdot (W_{2N,2N} \cdot (Y, Z))), \tag{2}$$

де $W_{2N,2N}$ та $W^*_{2N,2N}$ - матриці ДПФ та ОДПФ, які складаються з елементів $W_{2N}^{\langle k \cdot r \rangle_{2N}} = e^{-\frac{2\pi i}{2N} k \cdot r}$, $k, r = \overline{0,2N-1}$; Z складається з N нульових слів.

Лема 3. При множенні двох N-слівних чисел на основі ШПФ довжиною $2N$ кожен з N паралельних процесорів виконає $2 + 3\log_2 2N$ комплексних однослівних операцій множення та $2 + 6\log_2 2N$ комплексних однослівних операцій додавання/віднімання.

Доведення. У (2), для виконання одного ШПФ(ОШПФ) довжиною $2N$, необхідно виконати $\log_2 2N$ ітерацій ШПФ. На кожній ітерації «метелика»[1] виконується N комплексних множень, результати яких додаються/віднімаються до/з $2N$ комплексних елементів. Загалом, кожен паралельний процесор виконає $\log_2 2N$ комплексних операцій множення та $2\log_2 2N$ комплексних операцій додавання/віднімання при обчисленні одного ШПФ (ОШПФ). Усього необхідно виконати два ШПФ та одне ОШПФ. Також, кожному з N процесорів необхідно виконати додатково по дві поелементні комплексні операції множення. Лема доведена.

Представимо (1) для $j = 0$ у наступному вигляді з врахуванням (2):

$$R_{j=0} \leftarrow \sum_{i=0}^{M-1} \left(\frac{1}{2N} \cdot W^*_{2N,2N} \cdot ((W_{2N,2N} \cdot (X_i, Z)) \cdot (W_{2N,2N} \cdot (Y_i, Z)))\right).$$

У попередньому виразі множник $\frac{1}{2N} \cdot W^*_{2N,2N}$, що відповідає операції ОШПФ, можна винести за знак суми, що значно спрощує обчислення.

[1] «Метелик» – стандартна операція алгоритму ШПФ.

$$R_{j=0} \leftarrow \frac{1}{2N} \cdot W^{*}_{2N,2N} \cdot \sum_{i=0}^{M-1} ((W_{2N,2N} \cdot (X_i, Z)) \cdot (W_{2N,2N} \cdot (Y_i, Z))). \tag{3}$$

Окрім (3), можна отримати ще більше спрощення при реалізації операції багаторозрядного множення на основі ШПФ при обчисленні (1), якщо передобчислити всі ДПФ $\hat{X}_j$, $\hat{Y}_j$, $j = \overline{0, M-1}$. Використовуючи передобчислені ДПФ $\hat{X}_j$, $\hat{Y}_j$, $j = \overline{0, M-1}$, можна швидко знайти суми добутків ДПФ $\hat{R}_j \leftarrow \sum_{i=0}^{M-1} (\hat{X}_i \cdot \hat{Y}_{\langle i+j \rangle_M})$, $j = \overline{0, M-1}$, а потім на їх основі обчислювати елементи ЦЗ $R_j \leftarrow \frac{1}{2N} \cdot W^{*}_{2N,2N} \cdot \hat{R}_j$, $j = \overline{0, M-1}$. На прикладі обчислення ЦЗ довжини чотири, пропонується одним з кроків виконувати наступне обчислення:

$\hat{X}_0$	$\hat{Y}_0$	$\hat{Y}_1$	$\hat{Y}_2$	$\hat{Y}_3$
$\hat{X}_1$	$\hat{Y}_1$	$\hat{Y}_2$	$\hat{Y}_3$	$\hat{Y}_0$
$\hat{X}_2$	$\hat{Y}_2$	$\hat{Y}_3$	$\hat{Y}_0$	$\hat{Y}_1$
$\hat{X}_3$	$\hat{Y}_3$	$\hat{Y}_0$	$\hat{Y}_1$	$\hat{Y}_2$
	$\hat{R}_0$	$\hat{R}_1$	$\hat{R}_2$	$\hat{R}_3$

Рис. 2. Обчислення ДПФ елементів ЦЗ довжиною у 4 точки

На рис. 2 $\hat{X}_0$, $\hat{X}_1$, $\hat{X}_2$, $\hat{X}_3$, $\hat{Y}_0$, $\hat{Y}_1$, $\hat{Y}_2$, $\hat{Y}_3$, $\hat{R}_0$, $\hat{R}_1$, $\hat{R}_2$, $\hat{R}_3$ є ДПФ відповідних елементів ЦЗ X_0, X_1, X_2, X_3, Y_0, Y_1, Y_2, Y_3, R_0, R_1, R_2, R_3 (див. рис. 1). У кожному стовпчику елементи $\hat{Y}_0$, $\hat{Y}_1$, $\hat{Y}_2$, $\hat{Y}_3$ циклічно повторюються. Тому замість 16 ШПФ, у яких використовуються $\hat{Y}_0$, $\hat{Y}_1$, $\hat{Y}_2$, $\hat{Y}_3$, їх достатньо передобчилити 1 раз. Таким чином, для обчислення ЦЗ (1) довжини 4 (рис. 1) на основі ШПФ (рис. 2) необхідно передобчислити 8 ДПФ $\hat{X}_j$, $\hat{Y}_j$, $j = \overline{0, 3}$, та обчислити 4 ОШПФ $R_j \leftarrow \frac{1}{2N} \cdot W^{*}_{2N,2N} \cdot \hat{R}_j$, $j = \overline{0, 3}$. Загалом, необхідно виконати 12 ШПФ(ОШПФ) замість 32 ШПФ (16 багаторозрядних множень) та 4 ОШПФ.

Розглянемо загальний випадок, коли необхідно обчислити M сум, у яких використовуються X_i, $Y_{\langle i+j \rangle_M}$, $i = \overline{0, M-1}$, $j = \overline{0, M-1}$:

$$R_j \leftarrow \frac{1}{2N} \cdot W^{*}_{2N,2N} \cdot \sum_{i=0}^{M-1} ((W_{2N,2N} \cdot (X_i, Z)) \cdot (W_{2N,2N} \cdot (Y_{\langle i+j \rangle_M}, Z))),$$

$$j = \overline{0, M-1}. \tag{4}$$

Лема 4. При знаходженні сум (4) співмножників N-слівних чисел на основі ШПФ довжини $2N$, кожен з MN паралельних процесорів виконає $2M + 3\log_2 2N$ комплексних однослівних операцій множення та $2M + 3 \cdot 2\log_2 2N$ комплексних однослівних операцій додавання/віднімання у послідовній моделі обчислень.

Доведення. У (4) необхідно передобчислити $2M$ ШПФ для X_i, Y_i, $i = \overline{0, M-1}$, та обчислити M ОШПФ довжини $2N$. Згідно з Лемою 3, при задіянні процесорів,

кожен з процесорів N при обчисленні одного ШПФ (ОШПФ) виконає $\log_2 2N$ комплексних операцій множення та $2\log_2 2N$ комплексних операцій додавання/віднімання. Також, кожен з N процесорів виконає додатково по два поелементних комплексних множення при реалізації багаторозрядної операції множення на основі ШПФ. Таких додаткових обчислень необхідно виконати M . Лема доведена.

Порівняння методів

Далі порівняємо складності обчислення ЦЗ (1) за кількістю однослівних операцій при реалізації багаторозрядної операції множення на основі методу множення «у стовпчик» (Лема 2) з реалізацією (4) на основі ШПФ (Лема 4) для одного процесора в багатопроцесорній моделі обчислень. Знайдемо агреговані коефіцієнти кількості операцій додавання/віднімання та множення.

Агрегований коефіцієнт для Леми 4 може бути виражений наступним чином:

$$O_4(M,N) = 4 \cdot O_4^*(M,N) + 2 \cdot (2 \cdot O_4^*(M,N) + O_4^{\pm}(M,N)), \qquad (5)$$

де $O_4^*(M,N)$, $O_4^{\pm}(M,N)$ - загальна кількість комплексних однослівних операцій множення та додавання/віднімання у Лемі 4.

Коефіцієнт 4 у доданку $4 \cdot O_4^*(M,N)$ та коефіцієнт 2 у доданку $2 \cdot O_4^*(M,N)$ у співвідношенні (5) позначають, що для реалізації однієї операції множення над комплексними числами необхідно чотири однослівні операції множення над дійсними числами (алгоритм Винограда дозволяє це зробити за три операції) та дві комплексні операції додавання. Двійка перед скобками у доданку $2 \cdot (2 \cdot O_4^*(M,N) + O_4^{\pm}(M,N))$ означає, що комплексні числа мають дійсну та уявну частини, тому для їх реалізації необхідно використати у два рази більшу кількість операцій над дійсними числами. Вважаємо, що однослівні операції додавання та множення над дійсними числами займають однаковий час за швидкодією. Після розкриття дужок, отримуємо наступний вираз:

$$O_4(M,N) = 8 \cdot O_4^*(M,N) + 2 \cdot O_4^{\pm}(M,N).$$

Після використання Леми 4, отримаємо наступне співвідношення:

$$O_5(M,N) = 8 \cdot (2M + 3\log_2 2N) + 2 \cdot (2M + 3 \cdot 2\log_2 2N).$$

Після спрощення, отримуємо наступний агрегований коефіцієнт:

$$O_4(M,N) = 20M + 36\log_2 2N.$$

Агрегований коефіцієнт Леми 2 дорівнює $O_2(M,N) = 3MN$, вважаючи, що час виконання операцій множення та додавання/віднімання на сучасних комп'ютерах однаковий.

Знайдемо умови, при яких запропонований метод на основі ШПФ (Лема 4) є більш ефективним, ніж метод, заснований на методі множення «у стовпчик» (Лема 2), тобто $O_4(M,N) < O_2(M,N)$.

$$k_{2/4} = \frac{O_2(M,N)}{O_4(M,N)} = \frac{3MN}{20M + 36\log_2 2N}.$$

Таблиця 1. Значення коефіцієнту прискорення $k_{2/4}(M,N)$

	N						
M	**4**	**8**	**16**	**32**	**64**	**128**	**256**
1	0,09375	0,14634	0,24000	0,40678	0,70588	**1,24675**	**2,23256**
2	0,16216	0,26087	0,43636	0,75000	**1,31507**	**2,34146**	**4,21978**
3	0,21429	0,35294	0,60000	**1,04348**	**1,84615**	**3,31034**	**6,00000**
4	0,25532	0,42857	0,73846	**1,29730**	**2,31325**	**4,17391**	**7,60396**
5	0,28846	0,49180	0,85714	**1,51899**	**2,72727**	**4,94845**	**9,05660**
6	0,31579	0,54545	0,96000	**1,71429**	**3,09677**	**5,64706**	**10,37838**
7	0,33871	0,59155	**1,05000**	**1,88764**	**3,42857**	**6,28037**	**11,58621**
8	0,35821	0,63158	**1,12941**	**2,04255**	**3,72816**	**6,85714**	**12,69421**
16	0,44860	0,82759	**1,53600**	**2,86567**	**5,37063**	**10,10526**	**19,08075**
32	0,51337	0,97959	**1,87317**	**3,58879**	**6,88789**	**13,24138**	**25,49378**
36	0,52174	1,00000	**1,92000**	**3,69231**	**7,11111**	**13,71429**	**26,48276**
37	0,52358	**1,00452**	**1,93043**	**3,71548**	**7,16129**	**13,82101**	**26,70677**
64	0,55331	**1,07865**	**2,10411**	**4,10695**	**8,02089**	**15,67347**	**30,64339**
128	0,57571	**1,13609**	**2,24234**	**4,42651**	**8,73969**	**17,25843**	**34,08599**
256	0,58761	**1,16717**	**2,31849**	**4,60570**	**9,14966**	**18,17751**	**36,11462**
512	0,59374	**1,18336**	**2,35854**	**4,70084**	**9,36942**	**18,67477**	**37,22226**
1024	0,59685	**1,19162**	**2,37909**	**4,74990**	**9,48331**	**18,93374**	**37,80196**

Аналізуючи табл. 1, бачимо, що при обчисленні циклічної згортки довжини M багаторозрядних чисел довжиною N слів, запропонований метод є ефективним, починаючи зі згорток, довжиною у $M \geq 37$ точок та багаторозрядних чисел з $N \geq 8$ (у $32 \cdot 8 = 256$ бітів) слів на початку обчислень.

Цікаво, що коефіцієнт прискорення у паралельній моделі повністю співпадає з коефіцієнтом прискорення у послідовній моделі за однакових умов (однакових M , N). Зазвичай, коефіцієнти прискорення для різних моделей обчислень відрізняються.

Висновок

У даній роботі проаналізовано складність обчислення ЦЗ (циклічної згортки) багаторозрядних чисел за кількістю однослівних операцій (множення, додавання та віднімання) у паралельному модулі обчислень. На основі ЦЗ можна побудувати швидкий алгоритм багаторозрядної операції множення. Розглянуто два методи обчислення циклічної згортки з реалізацією багаторозрядної операції множення на основі множення «у стовпчик» та на основі ШПФ. Надано алгоритми реалізації операції множення на основі двох методів у паралельній моделі обчислень. Запропоновано швидкий метод на основі ШПФ, який дозволяє реалізувати обчислення циклічної згортки багаторозрядних чисел зі складністю, меншою, ніж складність на основі методу множення «у стовпчик». Розглянуто умови, при яких ШПФ є швидким. Показано, що використання ШПФ довжиною 16 є вже ефективним при обчисленні ЦЗ, починаючи з довжин згортки у 37 точок, де кожна точка є багаторозрядним числом у 8 слів (256 біт) на початку обчислень. Програмна реалізація на GPU (графічних прискорювачах) для ядра (паралельного процесора) на основі ШПФ довжиною 16 дозволяє виконати обчислення з використанням векторних операцій довжиною 16, що дозволяє будувати ще більш швидкі алгоритми.

При реалізації алгоритму на основі ШПФ, знаки переносу між машинними словами необхідно враховувати тільки на етапі обчислення ОШПФ, що є перевагою, у

порівнянні з реалізацією алгоритму на основі множення «у стовпчик», коли знаки переносу необхідно враховувати на всіх етапах обчислення ЦЗ багаторозрядних чисел. Це також дає перевагу при реалізації у паралельній моделі обчислень, де операції врахування знаку переносу можуть займати більший час, ніж всі інші операції загалом.

Довжини багаторозрядних множників при реалізації множення на основі ШПФ обмежені довжиною машинного слова, яку може обробляти процесор за одну команду. Запропонований метод дозволяє будувати алгоритми множення ще більших багаторозрядних чисел на основі ШПФ, яка використовується для обчислення циклічної згортки, що саме реалізує багаторозрядну операцію множення у паралельній моделі обчислень.

Література

1. David A. Pitassi. Fast convolution using the Walsh transform. – Applications of Walsh Functions, 1971, Proceeding, pp.130–133, April, 1971.
2. Davis W. F. A class of efficient convolution algorithms // Applicat. Walsh Functions. – 1972. – March. – pp.318–329.
3. Задірака В., Олексюк О. Комп'ютерна арифметика багаторозрядних чисел. – К.: Наук. думка, 2003. – 263 с.
4. Терещенко А. Н. Оптимизация метода Питасси вычисления свертки // Искусственный интеллект. – 2009. – № 1 – С. 204–212.
5. Терещенко А. Н., Мельникова С. С., Гнатив Л. А., Задирака В. К., Кошкина Н. В. Реализация операции умножения с использованием преобразования Уолша // Международный научно-технический журнал Проблемы управления и информатики. – 2010. – № 2. – С. 102–126.

Literatura

1. David A. Pitassi. Fast convolution using the Walsh transform. – Applications of Walsh Functions, 1971, Proceeding, pp.130–133, April, 1971.
2. Davis W. F. A class of efficient convolution algorithms // Applicat. Walsh Functions. – 1972. – March. – pp.318–329.
3. Zadiraka V., Oleksyk O. Multidigit computational arithmetic. – К.: Science opinion, 2003. – 263 p.
4. Tereshchenko A. N. Optimization of Pitassi method of convolution calculation // Artificial intellect. – 2009. – N 1 – P. 204–212.
5. Tereshchenko A. N., Melnikova S. S., Hnatyv L. A., Zadiraka V. K., Koshkina N. V. Calculation of multiplication, using walsh transform // Journal of automation and information sciences. – 2010. – N 2. – P. 102–126.

RESUME

A.M. Tereshchenko, V.K.Zadiraka

Fast proceeding of cyclic convolution of multidigit values based on FFT in parallel computational model

In this paper the complexity of calculation of the circular convolution of multidigit values in terms of single precision operations (multiplication, addition and subtraction) in parallel computational model is analyzed. Based on the circular convolution the fast algorithm of multidigit multiplication could be built. Two circular convolution calculation methods with the implementation of multidigit multiplication based on multiplication "in the column" and based on FFT e considered. Provided algorithms implement multidigit multiplication operation based on two methods in parallel computational model. A rapid method based on FFT calculation allows implementing the calculation of cyclic convolution of multidigit values with complexity less than complexity of method based on multiplication "in the column." The conditions under which the FFT is fast are considered. It is shown that using FFT length of 16 is effective in calculating circular convolution starting from the length of 37 points, where each point is a multidigit value of the length of 8 words (256 bits) in the beginning of computation. Software implementation on GPU based on FFT length of 16 allows using GPU vector operations of the length of 16 that speeds up algorithm sharply.

On implementing the algorithm based on FFT the carry flag between machine words is taken into account only at the stage of calculating IFFT, which is an advantage compared to the algorithm based on multiplication "in the column" when the carry flag is considered at all stages of computation. It also gives advantage of implementing on parallel computational model when operation for taking care of carry flag may take more time than all the other operations in general.

The lengths of multidigit multipliers in the implementation of multiplication based on FFT limited by the length of the machine word processor that can process in a single command. The proposed method broadens this restriction and allows building algorithms based on FFT for much bigger numbers.

Надійшла до редакції 04.09.2016

